REMARKS

In response to the final Office Action mailed on June 6, 2005, Applicant respectfully requests reconsideration of all rejections in the outstanding Office Action in view of the following remarks. Claims 1-18 as originally filed are currently pending.

I. The Obviousness Rejection Over Aho In View of Koizumi

Claims 1-18 stand rejected under 35 U.S.C. § 103(a), as allegedly unpatentable over Aho *et al.*, Compiler, Principles, Techniques, and Tools (1986) ("Aho") in view of U.S. Patent No. 5,586,323 to Koizumi *et al.* ("Koizumi"). *See* Office Action at page 4. Particularly, the Examiner contends that "Aho teaches all aspects of the applicant's claims but it does not specifically mention 'abstract register.'" *Id.* at page 6. In an attempt to cure this deficiency, Koizumi is introduced as allegedly teaching an "abstract register." *Id.* The Examiner then opines that "[i]t would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the intermediate representation of Aho with the abstract register taught by Kiozumi [*sic*], for the purpose of preserving a form of a program to be executed repeatedly." *Id.* (citations omitted). Applicant respectfully disagrees and traverses this rejection at least on the following grounds.

As stated in MPEP § 2143.01, to establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Applicant respectfully submits that Aho, either taken alone or in combination with Koizumi, fails to teach or suggest all of the limitations recited in the claims.

A. Aho Fails To Teach Or Suggest Either Step Recited In Claim 1

The Examiner begins with Aho and quotes particularly page 12, final paragraph with the lines:

> After syntax and semantic analysis, some compilers generate an explicit intermediate representation of the source program. We can think of this intermediate representation as a program for an abstract machine.

However, it is important to complete the quotation with the remainder of the paragraph:

> This intermediate representation should have two important properties; it should be easy to produce, and easy to translate into the target program.

This quote confirms that intermediate representation is an intermediate step between the source program (which is the starting point as input to the translator or compiler) and the target program (which is the output of the translator or compiler). This view is confirmed by Figure 1.9 of Aho and the related discussion on pages 10-11.

Claim 1 concerns the first stage, namely producing intermediate representation from a source program. Claim 1 is not concerned with the second stage, namely generating target code. We refer to the first words of claim 1 (emphasis added):

> A method of <u>generating an intermediate representation of program code</u> . . . .

The specific items in the recited method are set out in what the Examiner labels as steps (a) and (b) of claim 1, namely:

> (a) generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and
>
> (b) generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.

It is important to note that the method claim has elements, action and structure. However, contrary to the Examiner's assessment, there are differences over the cited prior art in all three areas and an obviousness-type rejection under 35 USC 103 is not appropriate.

The elements include "register objects representing abstract registers" and "expression objects each representing a different element of said program code."

The actions include "generating a plurality of register objects representing abstract registers ..." and "generating expression objects ... as that element arises in the program code."

The structure includes "a single register object representing a respective abstract register" and "each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects."

1. Aho fails to teach or suggest "generating ... register objects representing abstract registers"

To quote the Examiner in the Second Office Action, page 5:

> Registers are inherited [*sic*] from any microprocessor used to hold data for a particular purpose. In Aho's book, Aho further discloses the use of registers for any element of program code. In order for Aho to use those registers, they must be generated at some point.

Applicant respectfully submits that the compiler of Aho compiles from a high-level language down to a machine-oriented language. The high-level input language is not concerned with registers. Hence, the intermediate representation taught by Aho is not concerned with registers, but is instead concerned with higher-level concepts such as variables ("a", "b", "c") and operations ("*", "+", "-"). As confirmed by Chapter 9 of Aho concerning "Register Allocation": yes, explicit references to registers are generated "at some point," but this point is <u>after</u> intermediate representation has been produced.

2. Aho fails to teach or suggest "generating expression objects ..."

Step (b) of claim 1 generates expression objects, which are referenced either directly by the register objects of step (a), or indirectly through other expression objects. Note also that the expression objects are generated "as that element arises in the program code." If the register objects of step (a) are not disclosed by Aho, then it follows that the references of step (b) are also not taught by Aho.

The Examiner refers to page 49 of Aho:

> In Aho's book, page 49, under "Abstract and Concrete Syntax" section, "A useful starting point for thinking about ...". Here the node can be considered as a register object representing a respective abstract register and the operator is an expression object that is referenced by a register object (operand).

In reply: no, it can't. Page 49 is in Chapter 2 concerning "A Simple One-Pass Compiler." Page 49 is <u>irrelevant</u> to producing intermediate representation as in claim 1. The simple syntax tree discussed on page 49 shows how an input string (i.e., in a high level programming language) can

be divided into simpler components, and contrasts with a parse tree or concrete syntax tree which retains important semantic information. Syntactical analysis is one of the preliminary stages in a complier prior to producing intermediate representation.

3. Chapter 9 of Aho

Concerning steps (a) and (b), the Examiner refers to Chapter 9 of Aho, which discusses "Register Allocation" on page 517 and refers to "Rearranging the Order" on pages 558-559 with reference to Figures 9.18, 9.19 and 9.20. Unfortunately, the referenced sections on pages 517 and 558-9 are irrelevant to the generation of intermediate representation as set forth in claim 1.

Chapter 9 of Aho concerns code generation. Page 513, first paragraph, makes explicitly clear that intermediate representation has already been produced, and that the discussion and explanations in Chapter 9 only concern the generation of target code from the intermediate representation (emphasis added):

> The final phase in our compiler model is the code generator. It takes as input an intermediate representation of the source program and produces as output an equivalent target program, as indicated in Figure 9.1.

Page 517 and 558-9 relied upon by the Examiner are irrelevant to claim 1 and are explicitly taught as irrelevant on page 513.

The Examiner is instead referred to Chapter 8 of Aho, which discusses the generation of intermediate representation. However, none of the discussion in Aho relevant to producing intermediate representation, such as Chapter 8, discloses any of the elements, actions and structure of claimed method as recited in steps (a) and (b) of claim 1.

B. Koizumi Fails To Cure The Aho's Deficiencies

The Examiner already accepts that Aho does not teach "generating a plurality of register objects representing abstract registers" as in step (a) of claim1. Applicant respectfully submits that Aho also does not disclose step (b) of claim 1, as discussed above. Applicant respectfully does not accept that the skilled person would try to combine the respective teachings of Aho and Koizumi. Nevertheless, such a combination does not result in the claimed invention.

The Examiner refers to Koizumi, column 4, lines 53-58:

> ..., an abstract register machine (also referred to as ARM or Arm in abbreviation) having a plurality of registers is presumed, wherein an instruction sequence for the abstract register machine or ARM is made use of as a basic part of the common object program (referred to as the abstract object program).

However, the Examiner's interpretation of Koizumi is not correct. The Examiner selectively quotes column 4, lines 53-58. However, this selective quotation has to be read in context looking at the whole document, and we refer to the preceding paragraph at column 4, lines 34-50. Here, Koizumi teaches a three stage process:

> (1) a compiler; a subsystem for generating an object program (referred to as an "abstract object program") which is independent of the type of target machine;
>
> (2) a linker: a subsystem for linking together a plurality of abstract object programs; and
>
> (3) an installer: a subsystem for translating the abstract object program output by the linker into a machine language program for the target machine.

That is, Koizumi discloses a translator system for translating source programs into machine language programs. A source program 106 in Figure 1 of Koizumi is compiled using a compiler 1001 to produce an ArmCode program 1007. A plurality of ArmCode programs are combined using linker 1002 to provide a linked ArmCode program 1009. An installer 1003 then translates the linked ArmCode program 1009 into a machine language program 1011 executable by a computer 1012.

Koizumi first compiles a source program into abstract object programs (ArmCode) for an arbitrary virtual machine called an ARM machine, which is not restricted by physical constraints, and then "installs," i.e., does a second conversion, from the abstract object program into a machine language program for a real and physical target machine. Hence, Koizumi requires two separate translations: firstly from the source program to the abstract object program, and then secondly from the abstract object program to the target program.

1. "producing intermediate representation" in Koizumi

The compiler 1001 is discussed in more detail referring to Figure 2 of Koizumi from column 10, line 66 to column 11, line 21. The compiler takes as input the source program (e.g., source program 106 in Figure 1) and produces as output the abstract object program (ArmCode 1007) for the abstract register machine (ARM).

The compiler 1001 of Koizumi uses intermediate representation as an intermediate step between its input (i.e. the source program 106) and its output (i.e. the ArmCode program 1007). Koizumi explains the internal operations of the compiler 1001, at column 10, line 66 to column 11, line 16 (emphasis added):

> FIG. 2 is a diagram showing an exemplary structure of an abstract register machine compiler (generally denoted by 3100) according to the embodiment of the present invention. Referring to the figure, a source program 3102 undergoes at first a syntax analysis and a semantic analysis by a syntax/semantic analysis section 3104 of the compiler 3100 to be translated <u>into an intermediate language program 3106</u>, while information of a variety of symbols used in the source program 3102 is entered in a symbol table 3108. An intermediate language optimization section 3110 of the compiler 3100 performs a control flow analysis and a data flow analysis <u>on the intermediate language program 3106</u> to thereby realize a global optimization, as a result of which there is generated an ArmCode program 3112 as a sequence of abstract register machine code instructions prior to an optimization processing 3114. Subsequently, the optimization processing 3114 on the ArmCode program is carried out to generate an abstract object program 3116 as the optimized ArmCode instruction sequence.

Clearly, Koizumi does mention the use of intermediate representation ("an intermediate language program 3106") within the compiler 1001. However, it is just as clear that Koizumi does not disclose generation of this intermediate representation 3106 using any form of "register objects representing abstract registers," as in claim 1. The quoted passage is completely silent in this respect.

Later, Koizumi performs a second translation, from the ArmCode program 1007 into a machine language program 1011 for a target machine 1012. However, in Koizumi there is no mention of "intermediate representation" with reference to this second translation.

Applicant also respectfully points out that Koizumi simply sends the skilled person back to Aho for detailed information about the compiler 1001. *See* column 11, lines 16-22 (emphasis added):

> Concerning the methods of the syntax analysis, the semantic analysis, the control flow analysis, the data flow analysis and the global optimization, reference may be made <u>to "A. V. Aho</u>, R. Sethi and J. D. Ullman: Compilers, Principles, Techniques and Tools", Addison-Wesley Publishing Co., 1986, pp. 1-24.

2. The term "Abstract Registers" in Koizumi

It seems possible that the Examiner has been misdirected by use of the term "abstract register" in Koizumi. Nonetheless, this does not equate to "a plurality of register objects representing abstract registers" in "a method of generating intermediate representation," as in claim 1.

In Koizumi, the second translation "installs" the ArmCode 1007 as a machine language program 1011 for a target machine 1012. Part of this second translation is to allocate physical registers in the target machine 1012 to specific code instructions in the final executable program 1011. Like Chapter 9 of Aho discussed above, this register allocation function during code generation is irrelevant to the claimed invention.

Figure 32 of Koizumi shows how the "abstract registers" of the ARM are used in this second translation for correspondence with physical registers 4622 of the target machine A or B. The abstract registers 4626 of the abstract register machine will eventually become associated with physical registers 4622 in machine A and B through operations of the installer 2940. However, these abstract registers 4620 of Koizumi are quite unlike the "plurality of register objects representing abstract registers" provided in a method of generating intermediate representation as in claim 1. See column 27, lines 9-32:

> FIG. 32 is a view showing a part of correspondence relations between the physical registers incorporated in the machine B and the abstract registers. Although the abstract registers 4620 may be used in an arbitrary number as in the case of the machine A, the number of the physical register of the machine B is limited up to thirty-two. The abstract registers 4624 for placing function values therein are represented by "Funcval0" and "Funcval1" and correspondences are established to the physical register "8" and "9", respectively, as indicated by 4631 and 4632. Representing the abstract registers 4625 used for placing parameters therein by "Param0", "Param1", . . . , "Pramp", there are established correspondence relations between these abstract registers and the physical registers "24" to "29" as indicated by 4633 and 4634. The abstract registers 4626 used for arithmetic operation and represented by "Rr0", "Ar1", . . . , "Arq" bear correspondence relations to the physical registers "1" to "7" and "16" to "23", as indicated by 4635, 4636, 3637 and 4638, respectively. Similarly, the abstract registers 4627 used for the address calculation and represented by "Addr0", "Addr1", . . . , "Addrr", respectively, bear correspondence relations to the physical registers "1" to "7" and "16" to "23", as indicated by 4639, 4640, 4641 and 4642, respectively.

C. Summary

In summary, the Examiner accepts that Aho alone does not teach the claimed invention. When considering a "method of producing intermediate representation" as in claim 1, Aho in view of Koizumi does not arrive at the claimed invention. There is no teaching in Koizumi of any of the elements, actions and structures of steps (a) and (b) of claim 1. There is no additional information in Koizumi which the skilled person can use to supplement the teachings of Aho. As to intermediate representation, Koizumi, when read properly through the eyes of the skilled person, simply directs the skilled person back to Aho.

The same arguments also hold true for the other independent claims 16, 17, and 18. Because all the claim limitations in the independent claims are not taught or suggested by the prior art, all claims dependent therefrom, *e.g.*, claims 2-15, are also nonobvious.

D. Dependent Claims Recite Additional Novel and Nonobvious Limitations

Although dependent claims 2-15 are allowable at least by virtue of their dependency on independent claim 1, these claims recite additional subject matter which is not suggested by the cited art taken either alone or in combination.

For instance, concerning dependent claim 2, the compiler of Aho and the compiler system of Koizumi each receive a source program written in a high level computing language which is compiled into a machine executable language. By contrast, claim 2 recites that the program code (i.e. the source program) is expressed in terms of an instruction set of a subject processor. Neither of Aho or Koizumi discusses producing an intermediate representation of an executable source program as in claim 2.

Claim 3 recites that the register objects represent abstract registers corresponding to registers of said subject processor. Again, as discussed above there is no teaching of the "register objects" referred to in claim 3. Further, there is no teaching that any such register objects are register objects representing registers of a subject processor.

The other dependent claims are all allowable for like reasons, and also referring to the arguments made in response to the first Office Action. No further comment is believed to be needed here.

Applicant respectfully requests the Examiner to withdraw the rejection of claims 1-18.

II. Conclusion

In view of the foregoing, it is respectfully submitted that the present application is in condition for allowance, and an early indication of the same is courteously solicited. The Examiner is respectfully requested to contact the undersigned by telephone at the below listed telephone number, in order to expedite resolution of any issues and to expedite passage of the present application to issue, if any comments, questions, or suggestions arise in connection with the present application.

Applicant is concurrently filing herewith a Petition for a Two-Month Extension of Time, along with the requisite fee. In the event that a variance exists between the amount tendered and that required by the U.S. Patent and Trademark Office requires to enter and consider this Reply, or to prevent abandonment of the present application, please charge or credit such variance to the undersigned's Deposit Account No. 50-2613 (Order No. 45256.00003.CIP1).

Respectfully submitted,

November 7, 2005

By:

Trevor Q. Coddington, Ph.D.
Registration No. 46,633

PAUL, HASTINGS, JANOFSKY & WALKER LLP
Customer Number: 36183
P.O. Box 919092
San Diego, CA 92191-9092
Telephone: (858) 720-2500
Facsimile: (858) 720-2555